KONGERIKET NORGE
The Kingdom of Norway

Bekreftelse på patentsøknad nr
Certification of patent application no

2002 3653

Det bekreftes herved at vedheftede dokument er nøyaktig utskrift/kopi av ovennevnte søknad, som opprinnelig inngitt 2002.07.31

It is hereby certified that the annexed document is a true copy of the above-mentioned application, as originally filed on 2002.07.31

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

2003.08.22

Line Reum

Line Reum
Saksbehandler

PATENTSTYRET®
Styret for det industrielle rettsvern

BL. 130 - 5.000 - 01.2001 - Hegro Grafisk AS

Søker:	SimSurgery AS Sognsveien 75B 0855 Oslo
Fullmektig:	Tom Ekeberg c/o Mobile Media Company AS Karl Johans gate 8 0154 Oslo
Oppfinner:	Geir Westgaard Yvon Halbwachs
Oppfinnelsens tittel:	Method, system and computer program product for creating an irregular mesh description and an embedded geometric description in a computer graphics system

FIELD OF THE INVENTION

The present invention relates generally to computer graphics, and topological and geometrical representation of 3-dimensional objects. More particularly the invention relates to description of objects using meshes.

BACKGROUND OF THE INVENTION

Representations of 3-dimensional objects in computer graphics are often based on the creation of some kind of mesh consisting of vertices connected by edges. This mesh creates a frame upon which surfaces and texture can be superimposed. Such a model has two distinct, but interrelated, properties, namely the topology, which is a description of how the various vertices are connected by edges and faces, and the geometry, which is the position in space of the vertices and the shape of the surface superimposed on the mesh.

Boundary based representation of objects (B-rep) has been exploited for a number of years within such fields as Computer Aided Design (CAD), modeling of geological structures, and entertainment such as games and movies. B-reps represent objects by their boundaries. For example, a 3D cube is represented by its six faces, where the faces are tied together with some relationship information. An object is usually subdivided further, such that both volumes, faces, edges and vertices are represented explicitly, along with the positional relationships between these entities. The various B-rep based topological representations differ in their level of subdivision, the relationships established between the topological elements, and how they distinguish between the topological model and the data embedded in this model, e.g. geometry.

Subdivision is a technique wherein a mesh is successively refined by repeated subdivision. A smooth surface is defined as the limit of a sequence of such refinements. Subdivision allows arbitrary topology, but the topology has not been explicitly defined in prior art solutions. One example of how topology has been handled is in the form of lists, and in order to e.g. find the neighbors of a given face in the mesh, it would be necessary to traverse the entire lists.

An alternative approach has been to use a regular mesh and associate a surface spline patch with each face. A refinement of this method, known as hierarchical B-splines, involves refining the mesh locally and adding smaller patches to the refined area. Again, this method lacks any explicit data model representing the topology.

Various representations can be chosen for the geometric entities embedded in the topological model. At a simplest level, straight lines are used to describe curves between vertices, and triangulation is used to create surfaces. However, it is desirable to represent free-form surfaces. A well-known way of doing this is to use surface splines.

A B-spline is a smooth free-form surface defined by a set of mxn control vertices. A complex surface can be constructed by creating a mesh of control vertices. In order to reduce the number of control vertices necessary for representing a surface with a number of local changes or deformations, a method of using hierarchical B-splines is known from David Forsey, Hierarchical B-splines, published on the Internet at http://www.cs.ubc.ca/nest/imager/contributions/forsey/dragon/hbsplines.html. This technique consists of adding a B-spline patch to the region around the deformation, necessitation the addition of control vertices only to this patch.

SUMMARY OF THE INVENTION

Common to the prior art solutions is that they do not provide an effective data structure that allows efficient interaction between objects, and dynamic change of topology and geometry as a result of such interactions. The present invention aims at providing a method and a data structure that facilitates fast and efficient local interaction between 3D-objects, and fast efficient local refinement or local deformation of the 3D-object descriptions.

The present invention is based on the use of generalized maps in order to describe the mesh topology, and on geometric description of the surface superimposed on this mesh.

Generalized maps (G-maps) is a type of generalized boundary representation model where the basic topological element of the topological map is the dart, a semi edge of a graph. A map is defined by a set of darts, D, and a set of involutions (α_0, α_1, ..., α_n). Each involution "α_i" describes the links between darts corresponding to a dimension "*i*". In 3-dimensional space, these relationships are used to subdivide an object into topological volumes (α_3), a volume into topological faces (α_2), a face into topological edges (α_1), and an edge into topological vertices (α_0).

Spatial representation of a 3-dimensional topological map consists in associating geometric entities with cells. Points in space are associated with vertices (0-cells), curve arcs are associated with edges (1-cells), surfaces are associated with faces (2-cells), and geometric volumes are associated with topological volumes (3-cells). These associations constitute the embedding of geometrical data in the topological map.

The n-dimensional G-map is expressed as

$G = (D, \alpha_0, \alpha_1, \ldots, \alpha_n)$

where D is a finite set of darts, and $\alpha_0 \ldots \alpha_n$ are $n+1$ involutions on D. An involution is a set of darts couples or singles. In the 3-dimensional case, the darts are used as follows:

α_0 creates edges by linking darts of adjacent vertices;
α_1 connects edges by linking darts that meet in common vertices;
α_2 connects surfaces by linking darts that constitute a common edge;
α_3 connects volumes along a common face by linking darts that constitute the common face.

According to a first aspect of the invention, there is provided a method for describing a 3D-object using a mesh based geometric model where the topology of the description is described using G-maps and the geometry is based on coordinates in space associated with the vertices of said mesh.

One way of creating the geometry description is to repeatedly apply a subdivision algorithm until a sufficiently smooth surface has been applied.

For an example of subdivision algorithms, reference is made to E. Catmull and J. Clark, Recursively generated B-spline surfaces on arbitrary topological surfaces, *Computer-Aided Design* 10(6):350-355, November 1978.

A preferred way of creating the geometric description, however, is based on the creation of a refined inner mesh by applying a subdivision algorithm once, and using the vertices of this second mesh as control points for surface patches associated with the first mesh.

In a preferred embodiment of the invention, this is done by dividing each face into quads by defining a point in the middle of the face and defining points that divide each edge in two. Drawing lines from the new face point and down to each new edge point will subdivide the face into only rectangular quads. The resulting quad structure will only consist of rectangular patches. A smooth surface can then be created e.g. by associating a b-spline patch with each quad.

In order to create local refinement or distortion of the described geometric model, each quad can be subdivided further. According to the invention this is handled by creating a new G-map description for the topology of the subdivided quads and linking this to the previous level in a hierarchical manner. Additional geometric patches are then associated with the quads of the new topological level in the same manner as the original patches were associated with the quads of the original topological level.

The various references between objects in the data structure can be implemented in a number of ways. According to a preferred embodiment, two darts that are linked by an α_1 involution will define one corner of a quad, and reference to that quad will then be made through these darts. In the same way, reference from one hierarchical level of the topology to another is made through darts linked by a γ link. Since the topological levels of this hierarchy will represent finer or coarser mesh structures, the reference from a dart on one level to a dart on a finer level will be referred to as γ_{fine}, while a reference from a dart on a finer level to a dart on a coarser level will be referred to as γ_{coarse}. It should be realized that while the α involutions allows navigation through one G-map structure, a γ link results in a move from one G-map to another.

In this manner it will be easy to navigate through the topological structure simply by applying the various α-involutions and check references to quads and/or their corresponding geometrical patches.

In addition to the method of creating a description of the topological and geometrical structure, the invention also covers the data structure resulting from this method.

Another aspect of the invention is a computer system programmed in a manner suitable for generating or changing a description or a data structure according to the invention based on suitable input data representing 3-dimensional objects.

Yet another aspect of the invention is a computer program product which, when installed on a suitable computer, enables the computer to generate a description or a data structure according to the invention, based on suitable input data representing 3-dimensional objects.

DESCRIPTION OF THE DRAWINGS

A preferred embodiment of the invention will now be described with reference to the drawings, where:

Figure 1 is an illustration of the subdivision of a model into cells;

Figure 2 is an illustration of a 2G-map and corresponding topology and possible geometry;

Figure 3 is an illustration of a 2G-map and corresponding topology and possible geometry, showing the 3D properties of the model;

Figure 4 shows a data structure holding the information of the G-map and the embedded geometry;

Figure 5 shows how geometry can be linked with the topology of the G-map shown in figure 2;

Figure 6 illustrates refined mesh, patch structure and patching submesh;

Figure 7 illustrates how an additional level of topology and geometry can be added in order to create local refinement;

Figure 8 illustrates an additional level of topology covering two patches of the original mesh;

Figure 9 illustrates various cases of available inner mesh control points;

Figure 10 shows refinement and control points of a regular mesh;

Figure 11 shows refinement and control points of an irregular mesh:

Figure 12 shows a data structure for several refinement levels of the topology and geometry;

Figure 13 lists input for a single level mesh;

Figure 14 lists input for a mesh where two patches have been subject to local refinement by the addition of one more level.

DETAILED DESCRIPTION OF THE INVENTION

The following description is a presentation of the principles of generalized maps (G-maps) as applied to the field of computer graphics. The presentation will be based on the theoretical structure of a G-map and the association between the G-map and an embedded geometry. Also, examples of how this structure can be implemented as a data structure in a computer graphics system. Those skilled in the art will realize that implementation details regarding the actual data structures are possible and are intended to be covered by the scope and spirit of the invention.

FIG. 1 illustrates how a 3D model can be divided into volumes, faces, edges and vertices. A 3D model 101 is subdivided into two volumes 102. These are again subdivided, each into six faces 103, representing each side of the two volumes 102. Then each face is subdivided into four edges 104. Finally, each edge is subdivided into two half-segments, each of which will be referred to as a dart 105. Each dart is associated with exactly one vertex, one edge, one face and one volume, also referred to as 0-cells, 1-cells, 2-cells and 3-cells respectively. It should, however, be noted that it is not necessary to represent all these cells explicitly in the data structure describing the G-map topology. As described below, the preferred embodiment of the invention does not include an explicit representation of edges. Edges are found by using functions operating on the darts. Likewise, the preferred

embodiment described herein does not include explicit representations of volumes. The invention does not, however, prevent such descriptions from being included.

A generalized map of dimension $n \geq 0$ is a graph $G = (D, \alpha_0, \alpha_1, \ldots, \alpha_n)$, that consists of a non-empty, finite set of darts, D, and n+1 involutions α_i on D. The involutions create associations between darts in a manner that define the topology of the graph. A more detailed description is given in Generalized Maps in Geological Modeling: Object-Oriented Design of Topological Kernels, by Yvon Halbwachs and Øyvind Hjelle, Oslo, Norway, which is hereby incorporated by reference.

A G-map may in principle have any number of dimensions, but in computer graphics the most relevant is the 3-dimensional case, and the following exemplary description will be limited to that. A person skilled in the art will realize that the invention may be modified to other numbers of dimensions if that should prove convenient as a design option.

FIG. 2 illustrates how the topology of three faces 201, 202, 203 is described using G-maps. The G-map, G, is a set of darts D and involutions α_i, in this case

$G = 2G\text{-map}(D, \alpha_0, \alpha_1, \alpha_2)$,

$D = \{ 1, 2, 3, \ldots, 24 \}$

$\alpha_0 = \{ (1, 2), (3, 4), (5, 6), (7, 8), (9, 10), (11, 12), (13, 14), (15, 16), (17, 18), (19, 20), (21, 22), (23, 24) \}$

$\alpha_1 = \{ (1, 10), (2, 3), (4, 5), (6, 7), (8, 9), (11, 18), (12, 13), (14, 15), (16, 17), (19, 24), (20, 21), (22, 23) \}$

$\alpha_2 = \{ (1, 18), (2, 17), (3), (4), (5), (6), (7), (8), (9, 20), (10, 19), (11, 24), (12, 23), (13), (14), (15), (16), (19), (21), (22) \}$.

The involutions are sets of couples and singles. A couple (d_i, d_j) in the set α_k is equivalent to $\alpha_k(d_i) = d_j$ and $\alpha_k(d_j) = d_i$, while a single (d) is equivalent to $\alpha_k(d) = d$.

As can be seen from FIG. 2, couples belonging to the α_0 involution join together two darts belonging to different vertices. Such a couple of darts then define an edge between two vertices of the graph. It can also be noted that since each dart is associated with only one face, the edge between two adjacent faces will be implicitly defined twice, except at the edge of a surface, where there is no bordering face.

Couples belonging to the α_1 involution join together two darts belonging to the same vertex. This is where two edges are joined together. It will be realized that by repeatedly invoking α_0 and α_1 involutions, it is possible to travel along the boundary of a face. A single in the α_1 set would represent an extreme vertex that defined the end of a line or edge.

Couples belonging to the α_2 involution join darts that belong to the same edge but to different faces. These couples define the relationship between adjacent faces. Invoking an α_2 involution makes it possible to go from one face to its neighbor. Singles in the α_2 set indicates that there are edges that do not separate two faces, in other words, edges along the boundary of the mesh. It will be realized that e.g. a 2-G-map representation of a sphere as connected triangles will not have any singles.

It would be possible, but not necessary, for a 3D representation, to include α_3 involutions in order to represent adjacent volumes. The couples belonging to the α_3 set would then represent darts that belong to the same face, but adjacent volumes. Singles in the α_3 set would represent darts around the boundary of faces that represent the surface of the 3D model. Returning to FIG. 1, the two volumes 102 would be joined together by α_3 couples along the edges of the two surfaces facing each other.

An orbit of a dart d is the set of all darts in a G-map that can be reached by applying a subset of the involutions in the G-map, starting with d. It has already been mentioned that repeated application of α_0 and α_1 will define the boundary of a face. This is referred to as the 2-orbit. For 2-G-maps, the 0-orbit (α_1, α_2) is the set of darts incident to a vertex, the 1-orbit (α_0, α_2) is the set of darts incident to an edge, and the 2-orbit (α_0, α_1) is the set of darts incident to a face.

Also shown is the topological representation 204 of the G-map, and a possible geometry 205. The geometry depends on the embedded geometric description, as described below.

FIG. 3 is another example of a simple volume with a topology that is described using G-maps. The topology comprises vertices 301, edges 302 and faces 303. In this example, the topology of the volume is that of a cube 304, while it is illustrated that the geometry could for instance actually be that of a sphere 305. Again it is shown how α_0 involutions link darts together to create the edges, α_1 involutions link edges together at each vertex of the mesh, and α_2 involutions tie various faces together by linking darts that belong to the same edges but adjacent faces. It should be noted that it would be possible to leave out one of the faces in order to describe an object with the topology of an open box with five sides. The α_2 involutions of

the darts along the rim of the opening would then refer to the dart itself, since there would be no adjacent face.

Reference is now made to FIG. 4, which illustrates in a UML diagram a data structure according to a preferred embodiment of the invention. In an object oriented programming language such as C++, the vertices, darts and faces would preferably be defined as classes of objects, while e.g. edges and volumes are defined only implicitly. Also, while some relationships are explicitly defined through the use of pointers, others are defined only indirectly and can be found by examining the value of several pointers. However, it would be possible to structure this differently. Other choices could be made for which definitions that should be made explicit and which should be implicit. Information that is presented here as included in one class of objects could be included in other classes of objects. Also, additional classes of objects that tie objects together could be included, so that e.g. pointers that here are presented as referencing other objects directly may do so only indirectly. It would even be possible to implement the data structure of the invention in a programming language that is not object oriented. The skilled person will realize that it is the exploitation of the G-map data structure in a mesh based geometry, rather than the expression of this data structure in any particular programming language, that represents one of the central ideas behind the invention.

For every G-map structure 401, there will be an arbitrary number of darts 402. As already described, the darts will reference each other by way of α-involutions. The α-involutions may be implemented as pointers in each dart-object in an object oriented programming language.

The geometry is embedded in the topology. According to a preferred embodiment, this is done by way of three additional objects: vertices 403, quads 404, and faces 405. A vertex 403 is the topological representation of a 0-orbit in the mesh. A vertex can be referred to by any number of darts, but each dart will refer to only one vertex. These references can be implemented as a pointer in each dart, pointing to the associated vertex, one or more pointers in each vertex, pointing to the associated darts, or both.

In the same way, a face is the topological representation of a single mesh cell. A face is surrounded by a loop of darts that are tied together by alternating α_0 and α_1-involutions. Again, each dart will be associated with only one face, while each face may be associated with an arbitrary number of darts. The pointers may again be implemented in the dart objects, in the face objects, or in both.

It should be noted that it is sufficient to include one pointer from a face object to one of the associated darts, since the remaining darts can be found as the referenced

dart's 2-orbit. In the same way it is sufficient to include only one pointer from a vertex to a dart, since the remaining darts can be found as the referenced dart's 0-orbit.

Quads represent rectangular sections of each face. As will be described in greater detail with reference to FIG. 5, there can be an arbitrary number of quads to a face, but there will be only one vertex and two darts for each quad. According to a preferred embodiment of the invention, there is no explicit reference between vertices and quads (or indeed between any of the embedded objects), but each quad refers explicitly to two darts (actually to one, the other is found through an α_1-involution), and each dart refers to one quad.

According to a preferred embodiment of the invention, an embedding 406 is a class from which the various embedded objects inherit certain characteristics. In addition, the embedded objects will have additional characteristics depending on what kind of objects they are. Each vertex 403 will be associated with a variable holding the coordinates of a point in space 407, each quad 404 will be associated with a description of a surface patch 408, and each face will be associated with a variable holding the coordinates of a center point 409 of the face. (According to a preferred embodiment it is the center point of a refined inner mesh that is stored as this center point 409, as will be described below.)

The various variables and functions that describe geometry are preferably included as native to the embedded objects. However, they could also, as a matter of design choice, be located in separate objects that are pointed to by the embeddings, or in any other manner.

It should be noted that in the case where the G-map topology is used to describe a mesh that will be subdivided repeatedly, and not associated with surface descriptions, the quad/patch and face/center point may be left out. One of the main advantages of the invention, i.e. the topology description that can be quickly navigated in order to implement local refinement or distortion, will still be achieved.

According to this exemplary embodiment, there are no objects that explicitly define edges. However, all the darts incident to an edge can be found as the 1-orbit of one of the darts incident to the edge. Also, according to this embodiment, there are no embedded objects that describe volumes, and there are no α-involutions above α_2. It should be noted, however, that the principles of the invention are not limited in this way, and that any number of dimensions could be included in principle.

FIG. 5 shows in more detail how quads are derived from faces and associated with darts. In a first level of detail 501, the mesh of FIG. 2 is shown. In a second level of detail 502, the darts associated with one of the vertices of this mesh are shown,

along with the quads associated with these darts. Finally, in a third level of detail 503, two darts are illustrated along with one quad, and the face of which it is part.

The quads of a face are, according to a preferred embodiment of the invention, found using midpoint refinement. This method is described with reference to FIG. 6 below. It should be noted that if the embedded geometry is described using subdivision, quads are not necessary.

Each quad will be associated with one or both of the darts, as already described. It is not necessary to associate the quad with the face, since they will be linked through a common dart.

It should be noted that even if the face is linked to only one of the surrounding darts by a pointer in the face object, all the associated quads can be found by finding the 2-orbit of the dart pointed to by the face, and collecting all the quads pointed to by the darts in this orbit.

FIG. 6 illustrates in greater detail how the quads are constructed, and how a refined mesh is constructed from the first mesh.

According to a preferred embodiment of the invention, the quads are constructed using midpoint refinement. This is a method that is well known in the art, reference is made to Jörg Peters, Biquadratic C^1-surface splines over irregular meshes, Computer Aided Design Volume 27 Number 12, December 1995, pp. 895-903, 1995. Midpoint refinement is based on inserting points at the centroid of each mesh cell and the midpoint of each cell edge. This results in a refined structure of quadrilateral sub cells, and these are what we refer to as quads. Starting with the mesh 601, we get the patch structure 602. It should be noted that centroid and midpoint do refer to the geometric midpoints based on coordinates stored in the various vertices of the mesh, since the patch structure (or quad structure) is topological, and not expressly computed or constructed.

From the original mesh 601, a refined mesh 603 is created. This is done by applying a mesh refinement algorithm to the original mesh. One such algorithm is illustrated in FIG. 6. For every face in the original mesh a new vertex is found based on each original vertex and the two neighboring vertices. The coordinates (positions) to be associated with each vertex in the new mesh is then found as

$$V_0 = V + u*V_U + v*V_V + V_U \Lambda V_V$$

Where

u and v are constants

$V_U = V_R - V$

$V_V = V_L - V$

and V, V_L and V_R are the positions associated with the original vertex, and the vertices to the left and right of this in the original mesh, respectively.

The resulting vertices of the refined mesh are used as control points when the shape of each patch is found. However, the refined mesh will not provide all the necessary control points for constructing a surface spline patch. This is illustrated in a detailed view 604 of the patching submesh. The Vertex V_0 gives the control point C_0, and other vertices of the refined mesh found in the same way but based on other vertices of the original mesh, yield the control points C_1, C_2, C_3, C_5, C_6, and C_7. Preferably, C_8 is the average of the points surrounding the face of the inner mesh, and according to a preferred embodiment of the invention, this is also the point that is stored as the center point of the face described above. In the same way, C_4 can be found as the average of the inner mesh vertices that have been based on one point V of the outer mesh.

Turning now to FIG. 7, local refinement of the model will be described. The description is based on the preferred embodiment using midpoint refinement in order to find a patch structure and creating surface spline patches. However, other embodiments are within the scope of the invention, e.g. using subdivision in order to create the smooth surfaces. Subdivision could e.g. be used independently for the coarse mesh and for the local refined mesh.

Again the mesh of FIG. 2 is shown at a first level of detail 701, while at a second level of detail 702, a pair of darts and an associated quad are shown. This quad is again subdivided into a second level of detail, and on this second level a new G-map is created.

The first level G-map, or level 0, comprises a dart_level_0 and a quad_level_0, with all the properties already described. The second level G-map, level 1, comprises a vertex somewhere inside the quad at quad_level_0. Eight darts are associated with this vertex. The darts themselves are associated with each other in pairs by α_1-involutions, and it is each such pair of darts on dart_level_1 that span out each new quad on quad_level_1. In addition, each quad is associated with neighboring quads through α_2-involutions that tie together neighboring darts.

The construction of the level 1 G-map is similar to the reasoning behind midpoint division, but a few differences should be noted. First of all, this is the actual construction of a new G-map, and instead of a theoretical centroid, an actual midpoint is computed and stored in a vertex that represents the center of the quad at level 0. This center can be computed based on the geometry embedded in the level 0 mesh. However, the reason for creating this second level of detail could be the result of interaction between several objects. If e.g. an object is hit by another

object somewhere inside a quad, causing a deformation, the point of impact could be used as the vertex for this second level G-map. Indeed such an interaction between two objects could be the very reason why the level 1 G-map is constructed at all.

Second, it should be noted that as long as the refinement does not extend beyond the original quad, none of the darts on dart_level_1 will be linked with other darts by α_0-involutions, and hence there will be no complete 2-orbit and therefore no face associated with any of these darts. However, if the refinement does extend beyond one quad, a second level G-map will be created in the same way in the adjacent quads, and they will be linked with α_0-involutions across the quad boundaries, as shown in FIG. 8.

Finally it should be noted that it will be necessary to be able to navigate from a G-map on one level to a G-map on another level – finer or coarser. This is done by adding γ-links as pointers from darts on one dart level (belonging to one G-map) to associated darts on another dart level (belonging to another G-map). Preferably, this is done by adding pointers to the dart objects. One pointer, $\gamma_{\_fine}$, points to a dart on the next finer level, and one pointer, $\gamma_{\_coarse}$, points to a dart on a coarser level. It is not, however, necessary to add these pointers to every dart object. When going from a coarser level to a finer level, it is sufficient to have a pointer from one of the two darts that are associated with the patch that is refined. If one dart does not contain such a reference, the other dart, found by applying an α_1-involution, will. Similarly, when going from a finer to a coarser level, it is sufficient to search the darts of the 0-orbit around a vertex in order to find the dart that contains a reference to the coarser level.

FIG. 8 shows local refinement similar to that of FIG. 7, but where the refinement involves two adjacent patches of the first, coarser mesh. It can be seen that the exact same procedure is followed, with the only difference that darts are linked across patch boundaries. It will also be noticed that there are still no complete α_2-orbits, so the finer mesh does not include any faces.

FIG. 9 is an illustration of the various cases that may occur when constructing the surface spline patches form control points found in an inner mesh. As described with reference to FIG. 6, the control points will be found as vertices in the inner mesh, or as averages of these. However, in the case of local refinement, there will be a lot of border conditions where the G-map is incomplete, and where the inner mesh does not produce all these control points. However, since the aim is to construct smooth surfaces, the necessary information can be found from the coarser level G-map. FIG. 9 illustrates five different cases, where case 0 represents the complete situation already described. The other cases represent situations where

some of the control points cannot be found from the inner mesh of the G-map on the same level.

FIG. 10 illustrates the patch on a coarse level, and the refined mesh on a finer level in the case where the mesh is regular. In this case it is sufficient to create the surface spline patches with control points in each corner of the patch, halfway along each edge, and in the center of the patch. These B-control points can be derived from the C-control points of the inner mesh as shown in table 1 for the five cases illustrated in FIG. 9.

PATCHING REGULAR AREA QUADS

Control points	Case 0	Case 1	Case 2	Case 3	Case 4
B_{00}	C_4	C_4	C_4	C_4	C_4
B_{10}	$(C_0+C_5)/2$	$(C_0+C_5)/2$	$(C_0+C_5)/2$	$(C_0+C_5)/2$	$(C_0+C_5)/2$
B_{20}	$(C_0+C_5+C_6+C_7)/4$	–	$(C_0+C_5+C_6+C_7)/4$	–	$(C_0+C_5+C_6+C_7)/4$
B_{01}	$(C_0+C_3)/2$	$(C_0+C_3)/2$	$(C_0+C_3)/2$	$(C_0+C_3)/2$	$(C_0+C_3)/2$
B_{11}	C_0	C_0	C_0	C_0	C_0
B_{21}	$(C_0+C_7)/2$	–	$(C_0+C_7)/2$	–	$(C_0+C_7)/2$
B_{02}	$(C_0+C_1+C_2+C_3)/4$	–	–	$(C_0+C_1+C_2+C_3)/4$	$(C_0+C_1+C_2+C_3)/4$
B_{12}	$(C_0+C_1)/2$	–	–	$(C_0+C_1)/2$	$(C_0+C_1)/2$
B_{22}	C_8	–	–	–	–

Table 1.

As can bee seen from table 1, all points can be found only in case 0. In the remaining cases there is not enough information in the inner mesh constructed from the refined G-map. This means that it will be necessary to find this information from the coarser level, since the transition from the level 0 surface to the level 1 surface should be smooth. The patch edges that follow the patch edges of the coarser level are in FIG. 9 shown as bold lines. It is for these edges information will be found on the coarser level. Returning to FIG. 10, the B-control points will either be found directly as control points on the coarser level, in the case where control points on the two levels overlap, or they can be placed between two control points in a manner that is well known from Bezier and spline theory.

In areas where the mesh is irregular it is necessary to introduce more B-control points. FIG. 11 shows an example of how the pattern of control points could be in this case. The same principles apply in the sense that whenever possible, B-control points are derived from the C-control points of the inner mesh, while when this is not possible, the necessary information is found on the coarser leve.

Because of the ease with which the G-map structure can be navigated by way of α-involutions, and with which it is possible to go from one G-map to another through γ-links, the it is not necessary to search through all the information available in

order to find the control points. It is sufficient to travel through the refined G-map structure until the relevant γ-link is found, and travel along the relevant edges of this coarser G-map in order to find the right data.

Turning now to FIG. 12, a data structure corresponding with that in FIG. 4 is shown, illustrating how the various levels are linked. It can be seen that the links between objects on the same level (the same G-map) are the same as in FIG. 4, while additional γ-links to finer and coarser G-map levels are also included. For the finest and the coarsest level, the $\gamma_{\_finer}$ and $\gamma_{\_coarser}$ links, respectively, will be null pointers.

Finally FIG. 13 illustrates the input data that is used in order to create a single level G-map in accordance with the invention. The nodes are numbered, and for each node a coordinate is given. For the sake of simplicity, the example is two dimensional, giving only two coordinates for each node. Then the faces are numbered, and each node belonging to the face is listed. Finally the quads are listed, but only as parameters to be used when operating on the input data. For each quad the associated pair of node and face is given. Following that constants that represent blend ratios are given. These values are used when computing the vertices of the inner mesh. Finally the level of refinement is given. In this case all the quads belong to level one, since no refinement has taken place.

FIG. 14 shows a similar example, but here the model has been refined with an additional level. The nodes and faces are the same, but the list of quads for level 0 has been changed only in that there is an indication that this level has been refined on level 1. Following this is a list of quads for this new level. In this case, since it is quite possible that there will not be any complete faces on this level, and indeed there are none in this example, the quads do not refer to faces and nodes, but to the quad on the coarser level of which it is a refinement, and simply a number that is defined according to the sequence in which the quads can be found when following a 0-orbit around the refinement point starting with the dart with which the γ-links are associated. Other ways of enumerating the quads of a refined G-map level are possible.

The invention has been described in terms of a method for describing a data structure containing topological and geometric information. The method is primarily intended to be implemented on a computer system with the necessary hardware component to process and store this information and render it in the form of graphics on a display unit.

The invention may be implemented as computer software instructions stored on a computer readable medium, such as a hard drive in a computer system or in other memory on such a system, or on other computer readable medium such as a CD-

ROM, a disk or any other kind of magnetic or magneto-optical storage medium. The instructions comprising the computer program may also be carried on a propagated signal.

5

CLAIMS

1. Method for creating an irregular mesh description and an embedded geometric description in a computer graphics system, comprising the steps of:

receiving topological input data representing vertices and faces of the mesh,

creating a G-map representation of the topology of said mesh based on said input data,

associating coordinates in space with the vertices of said mesh, and

creating a smooth geometric description from said mesh and said coordinates.

2. Method according to claim 1, wherein the step of creating a geometric description comprises applying a subdivision algorithm repeatedly until a sufficiently smooth surface has been created.

3. Method according to claim 1, wherein the step of creating a geometric description comprises the step of creating a refined mesh based on said first mesh and said coordinates, and using coordinates associated with the vertices of said refined mesh as control points in order to create surface patches associated with said first mesh.

4. Method according to claim 3, wherein said refined mesh is created by applying a mesh refinement algorithm, and where each patch of said first mesh is created as a surface spline associated with a rectangular section of said first mesh.

5. Method according to one of the previous claims, wherein the step of creating a G-map comprises the steps of creating a set of darts each associated with one vertex and one face of said first mesh; and creating a number of involutions that establish associations between pairs of darts so that an α_0 involution links two darts associated with adjacent vertices but the same face, creating an edge, an α_1 involution links two darts associated with the same vertex and the same face, and an α_2 involution links two darts associated with the same vertex but adjacent faces, linking two adjacent faces.

6. Method according to one of the previous claims, wherein a local refinement of said first mesh is created by defining a second mesh corresponding with one or more rectangular sections of said first mesh, subdividing these sections into smaller rectangular sections and describing the topology of said second mesh with a second G-map representation.

7. Method according to claim 6, wherein said second G-map structure is linked to said first G-map structure through γ-links between darts on the different levels.

8. 8. Computer system for creating an irregular mesh description and an embedded geometric description from input data, comprising:

input interface for receiving topological input data representing vertices and faces of the mesh,

processing means for creating a G-map representation of the topology of said mesh based on said input data and storing said representation in memory,

processing means for associating coordinates in space with the vertices of said mesh and storing said coordinates in memory,

processing means for creating a smooth geometric description from said mesh and said coordinates, and

output interface for outputting said smooth geometric description for representation on a display.

9. Computer system according to claim 8, wherein said means for creating a geometric description comprises processing means for applying a subdivision algorithm repeatedly until a sufficiently smooth surface has been created.

10. Computer system according to claim 8, wherein said means for creating a geometric description further comprises processing means for creating a refined mesh based on said first mesh and said coordinates, and using coordinates associated with the vertices of said refined mesh as control points in order to create surface patches associated with said first mesh.

11. Computer system according to claim 10, wherein said means for creating said refined mesh is capable of applying a mesh refinement algorithm, and of creating each patch of said first mesh as a surface spline associated with a rectangular section of said first mesh.

12. Computer system according to one of the claims 8 to 11, wherein said means for creating a G-map comprises means for creating a set of darts each associated with one vertex and one face of said first mesh; and means for creating a number of involutions that establish associations between pairs of darts so that an α_0 involution links two darts associated with adjacent vertices but the same face, creating an edge, an α_1 involution links two darts associated with the same vertex and the same face, and an α_2 involution links two darts associated with the same vertex but adjacent faces, linking two adjacent faces.

13. Computer system according to one of the claims 8 to 12, further comprising processing means for creating a local refinement of said first mesh by defining a second mesh corresponding with one or more rectangular sections of said first mesh, subdividing these sections into smaller rectangular sections and describing the topology of said second mesh with a second G-map representation.

14. Computer system according to claim 13, further comprising means for creating a link between said second G-map structure and said first G-map structure through links between darts on the different levels and storing these links in memory in a way that is associated with one or both G-map structures.

15. Computer system according to one of the claims 8 to 14, wherein the various processing means comprises a combination of computer program instructions and general purpose hardware.

16. Computer system according to claim 15, wherein said computer program instructions are stored on a persistent memory device in said computer system.

17. Computer program product comprising computer instructions that, when installed on a computer, makes said computer capable of performing the method of one of the claims 1 to 7.

18. Computer program product according to claim 17, stored on a computer readable medium.

19. Computer program product according to claim 18, wherein said computer readable medium is a CD-ROM or DVD-ROM.

20. Computer program product according to claim 18, wherein said computer readable medium is a magnetic or magneto-optical storage medium.

21. Computer program product according to claim 17, carried on a propagated signal.

ABSTRACT

The present invention relates to a method for creating an irregular mesh description and an embedded geometric description in a computer graphics system. The method comprises steps of receiving topological input data representing vertices and faces of the mesh, creating a G-map representation of the topology of said mesh based on said input data, associating coordinates in space with the vertices of said mesh, and creating a smooth geometric description from said mesh and said coordinates.

The invention also includes a computer system capable of performing the method, as well as computer program product that, when installed on a computer system, enables such a system to perform the invention.

(Fig. 4)

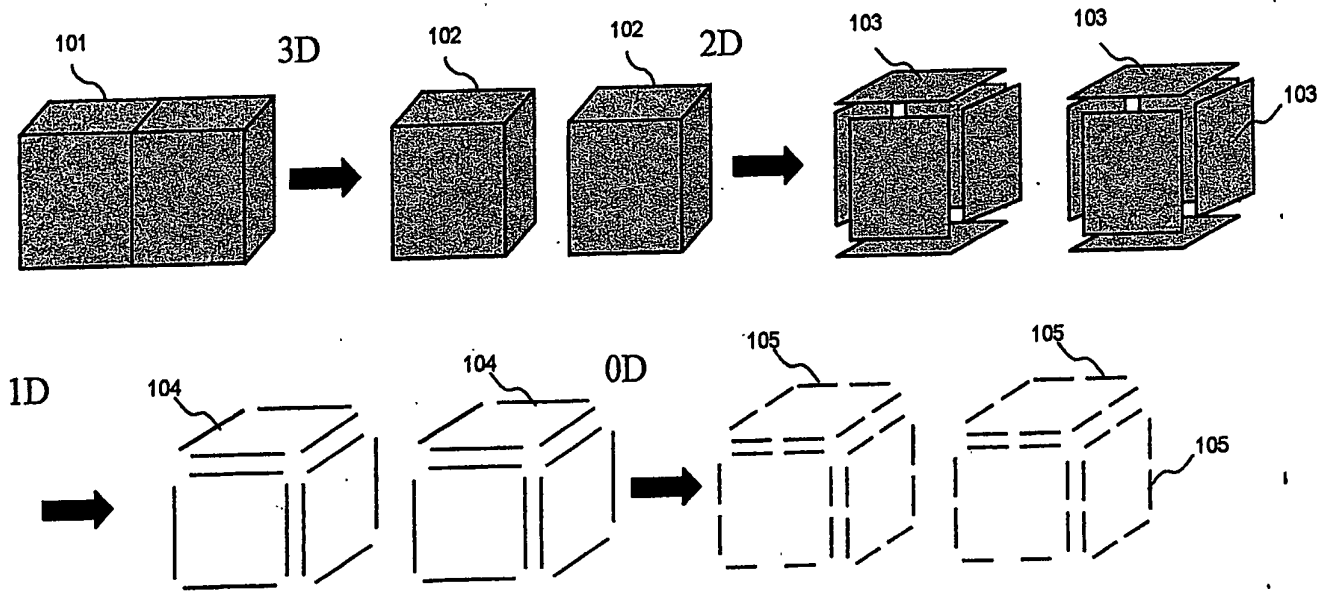
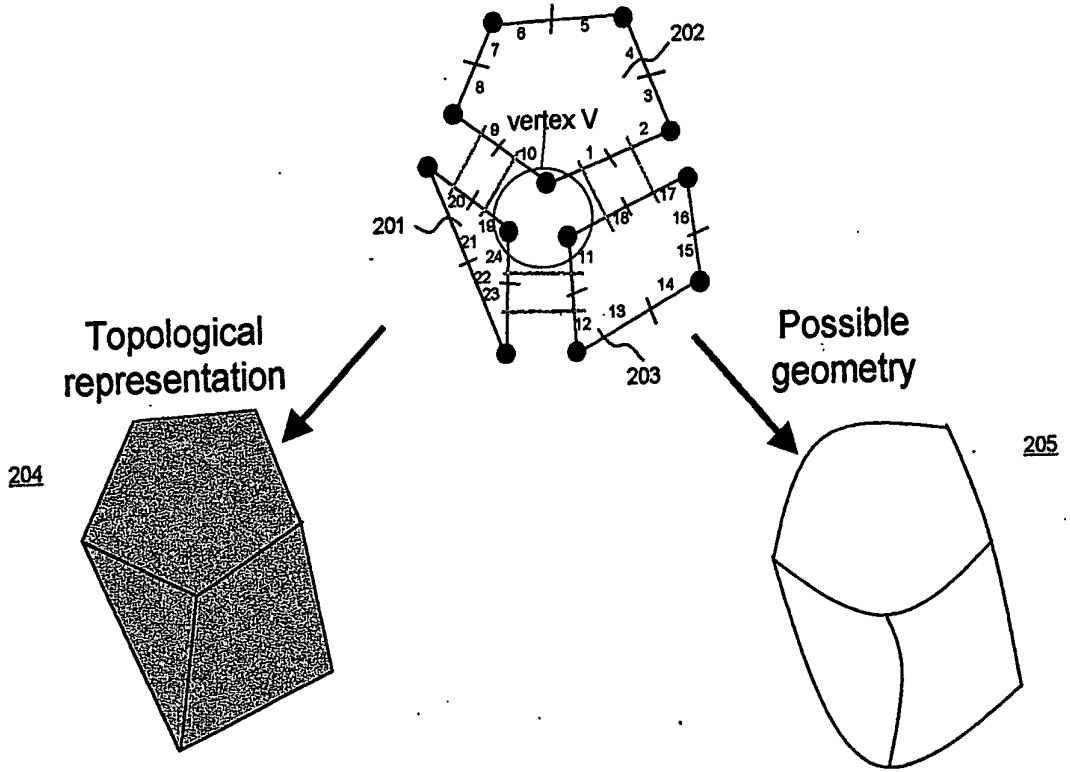


Fig. 1



α_0 : G = 2G-map(D, α_0, α_1, α_2), D = { 1, 2, 3, ..., 24 }

α_1 : α_0 = { (1, 2), (3, 4), (5, 6), (7, 8), (9, 10), (11, 12), (13, 14), (15, 16), (17, 18), (19, 20), (21, 22), (23, 24) }

α_2 : α_1 = { (1, 10), (2, 3), (4, 5), (6, 7), (8, 9), (11, 18), (12, 13), (14, 15), (16, 17), (19, 24), (20, 21), (22, 23) }

α_2 = { (1, 18), (2, 17), (3), (4), (5), (6), (7), (8), (9, 20), (10, 19), (11, 24), (12, 23), (13), (14), (15), (16), (19), (21), (22) }

Fig. 2

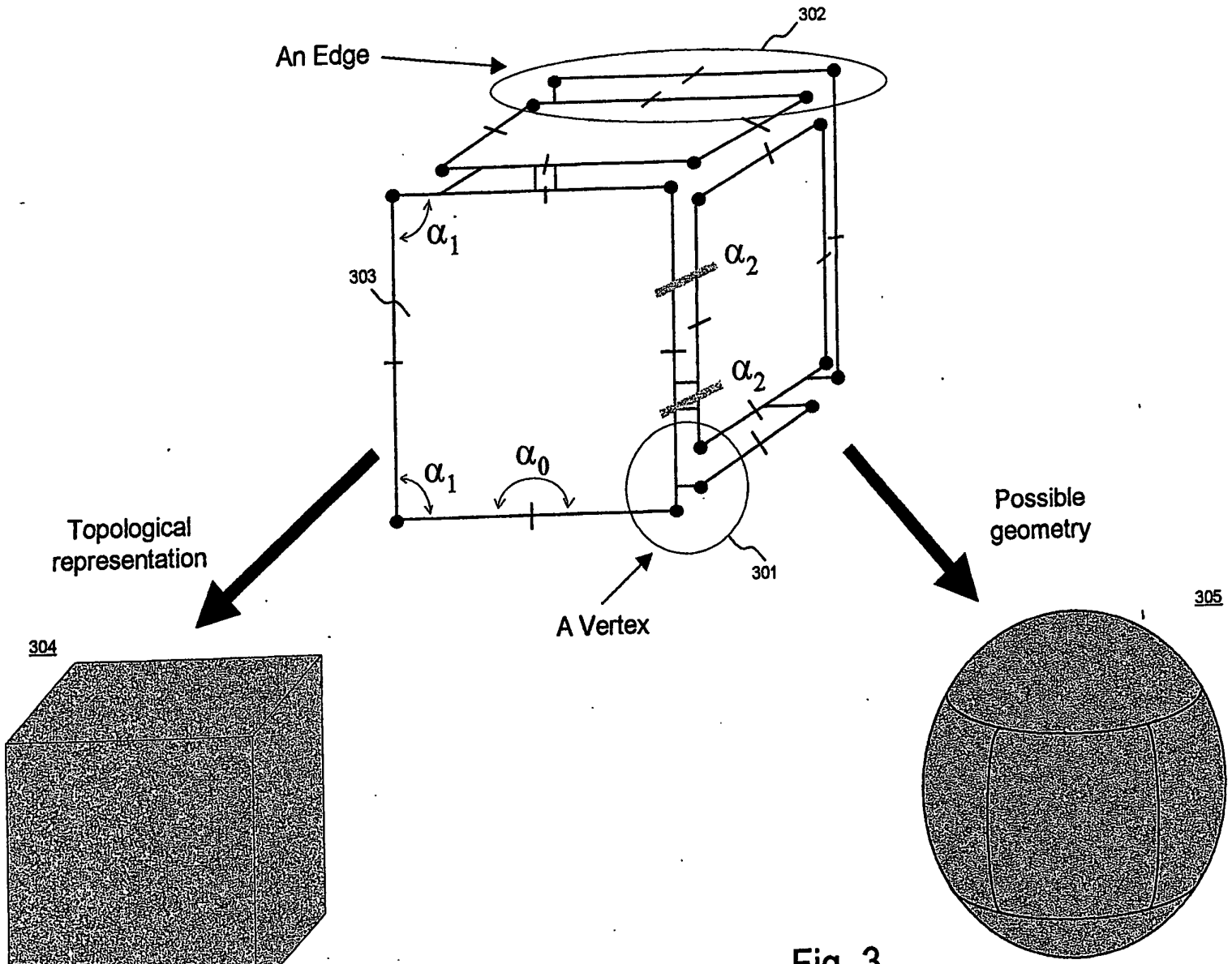

An Edge
302
α_1
303
α_2
α_2
α_1
α_0
301
A Vertex
Topological representation
304
Possible geometry
305

Fig. 3

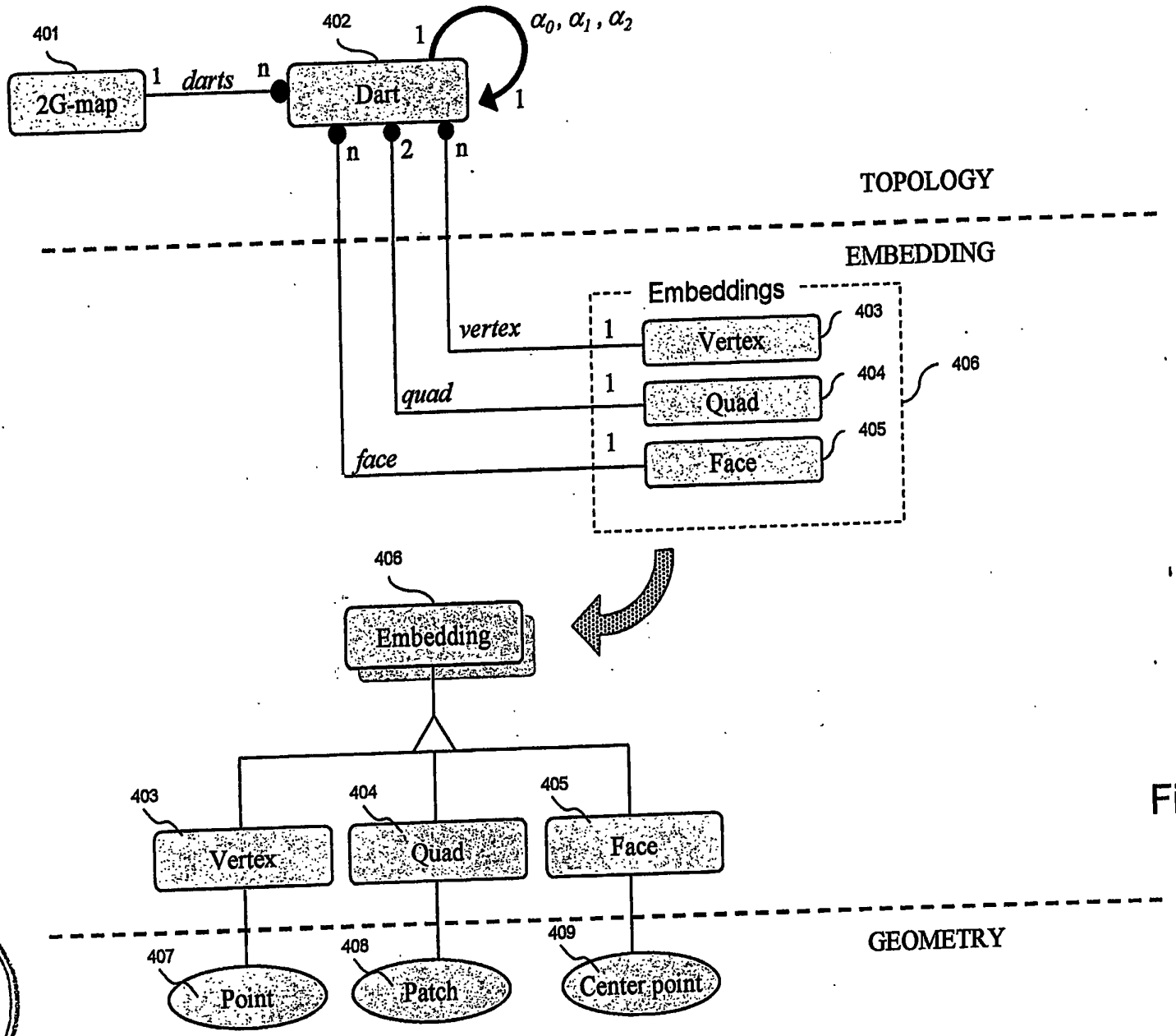
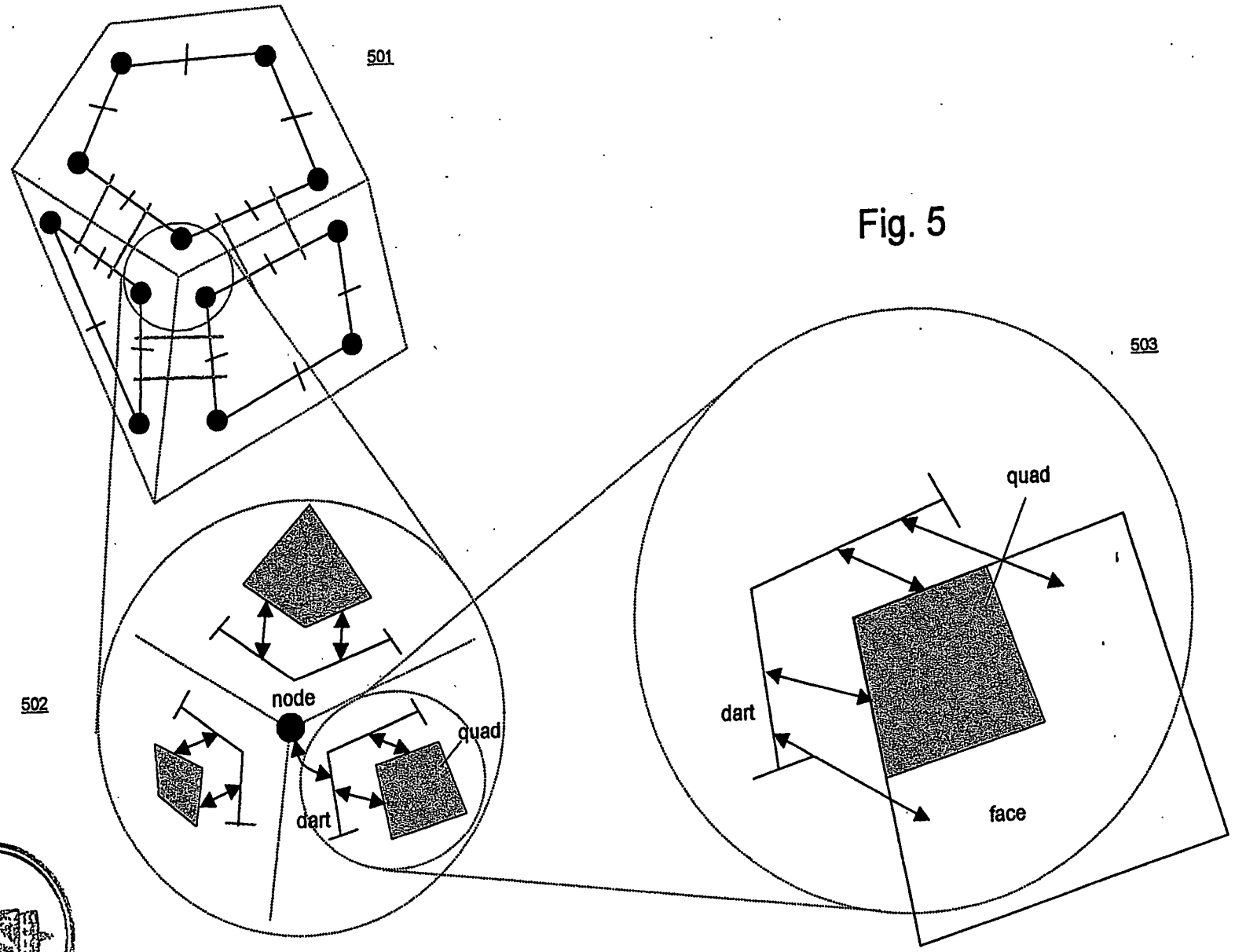

401
2G-map
1
darts
n
402
Dart
1
$\alpha_0, \alpha_1, \alpha_2$
1
n
2
n
TOPOLOGY
EMBEDDING
Embeddings
vertex
1
403
Vertex
quad
1
404
Quad
406
face
1
405
Face
406
Embedding
403
Vertex
404
Quad
405
Face
GEOMETRY
407
Point
408
Patch
409
Center point

Fig. 4

Fig. 5



Mesh
601
V_L
V_R
V
Mesh refinement
Refined mesh
patch
603
V_0
$V_U = V_R - V$
$V_V = V_L - V$
$V_0 = V + u*V_U + v*V_V + w*V_U \wedge V_V$
Patch structure
quad
602
Patching submesh
C_8
patch
C_1
C_7
C_0
C_2
C_6
C_3
C_4
C_5
604

Fig. 6

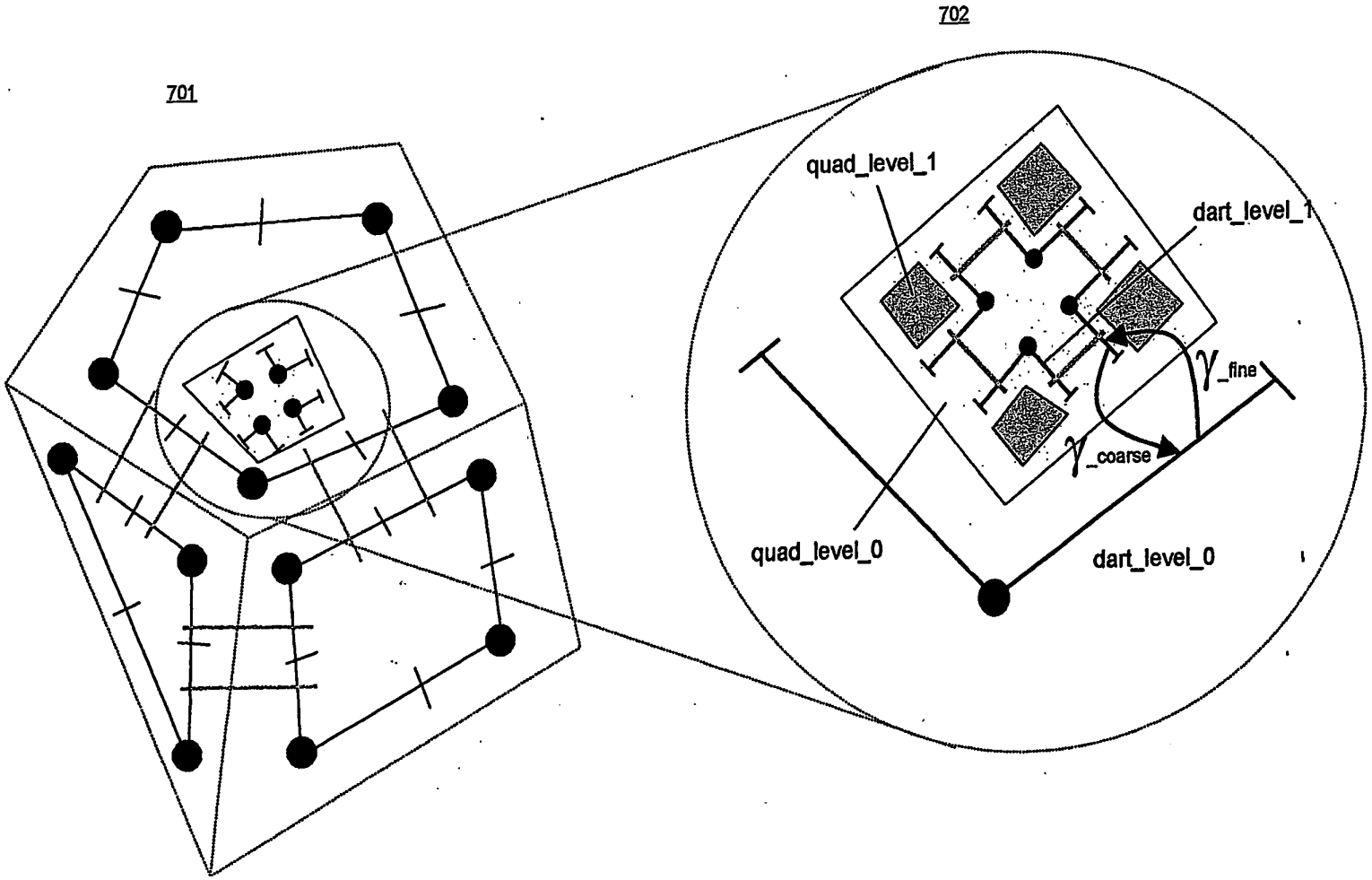


Fig. 7

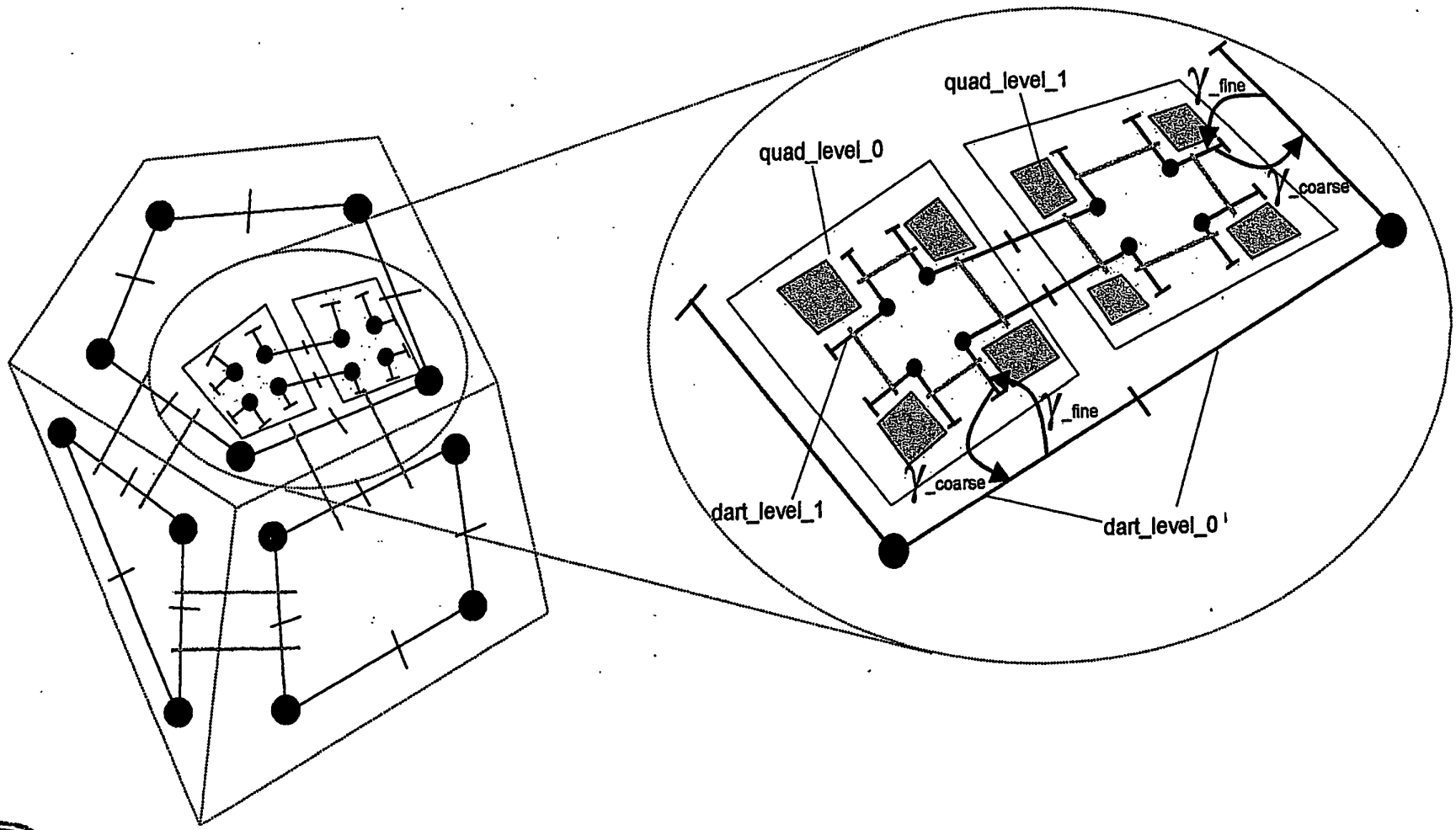
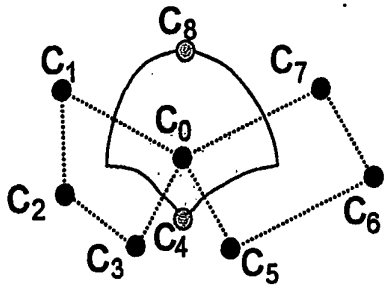
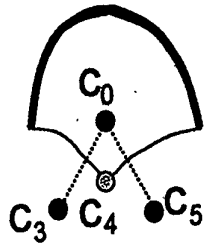


Fig. 8

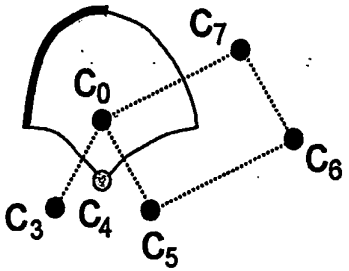
Case 0



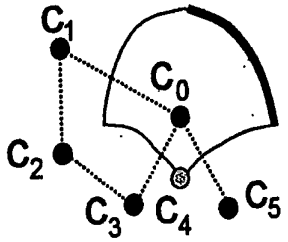
Case 1



Case 2



Case 3



Case 4

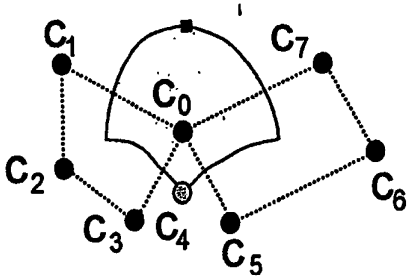


Fig. 9

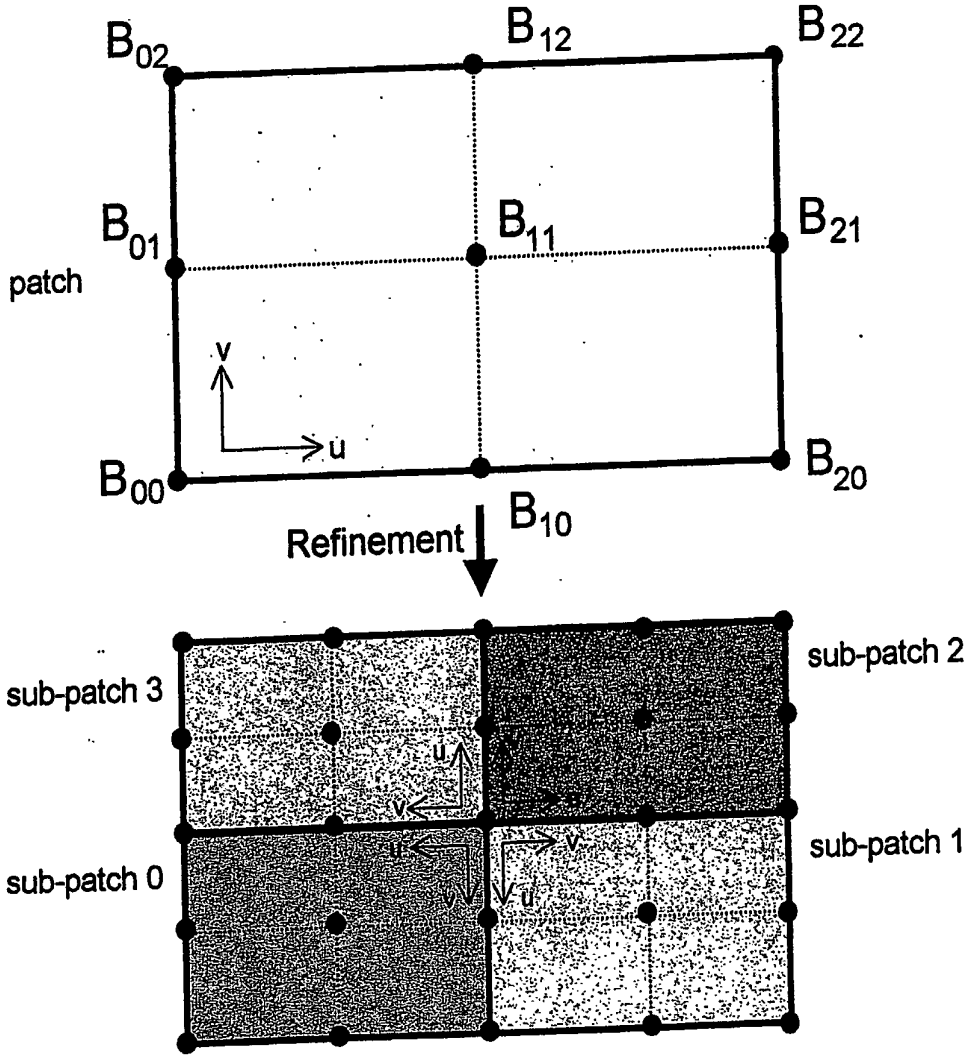


Fig. 10

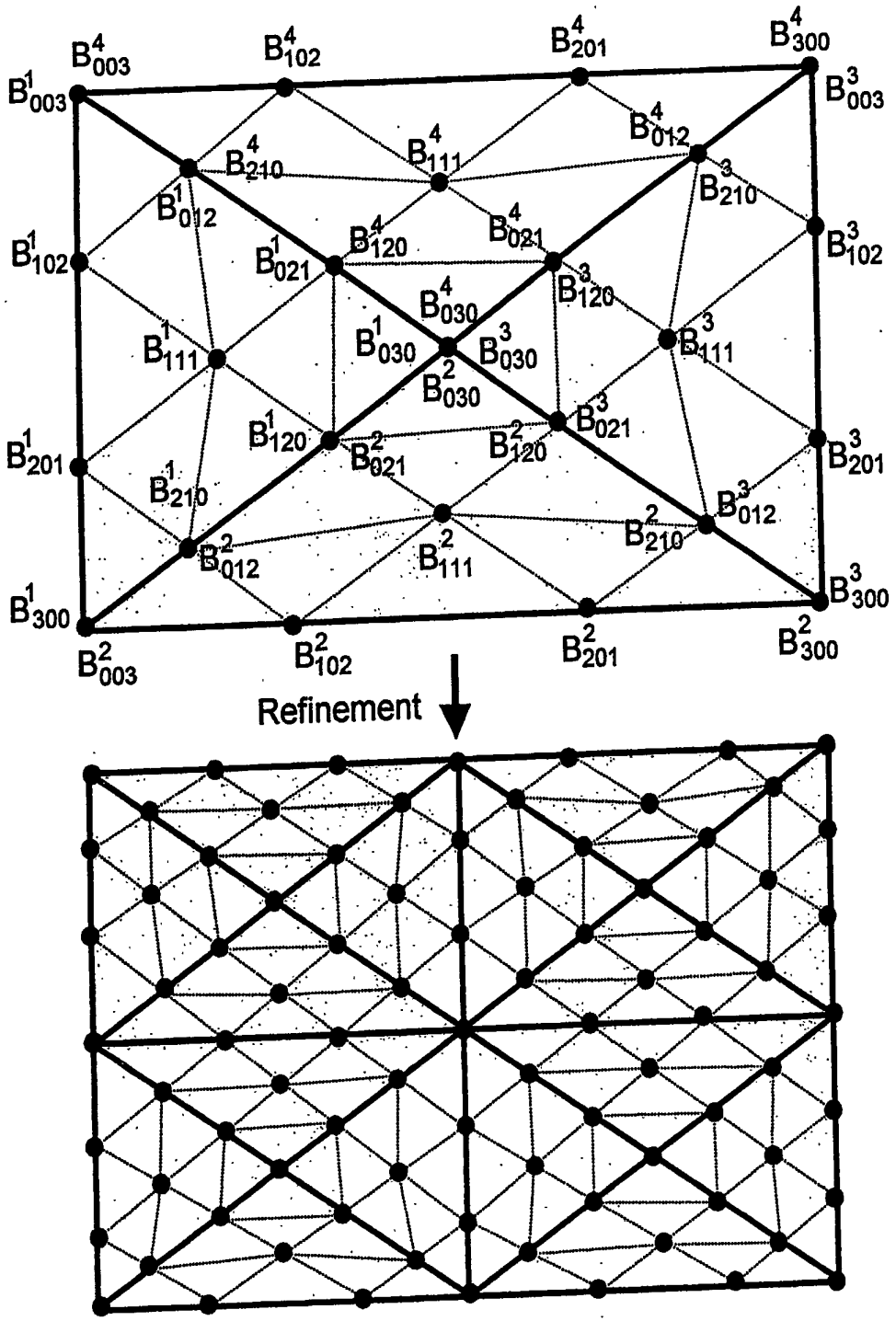


Fig. 11

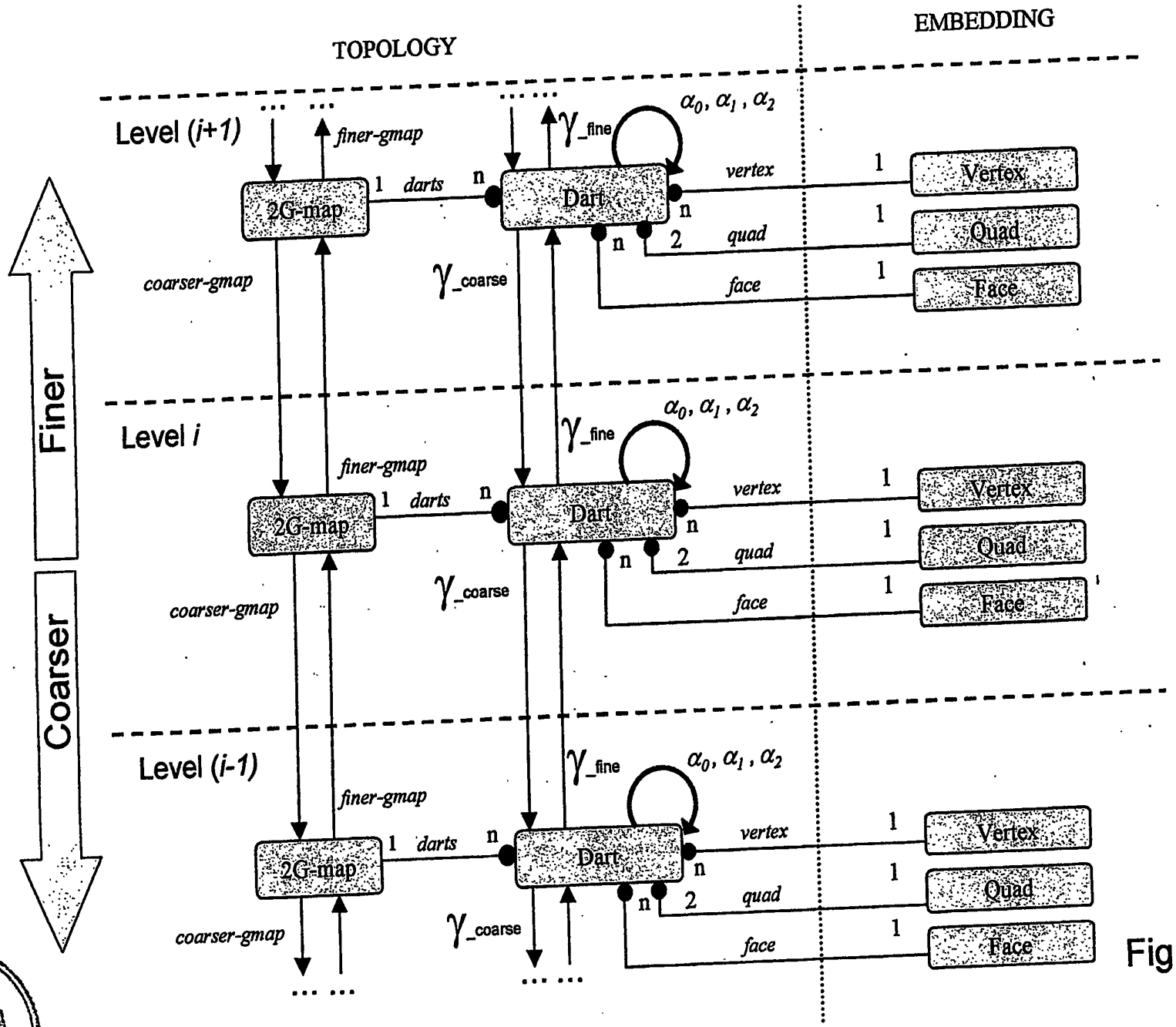
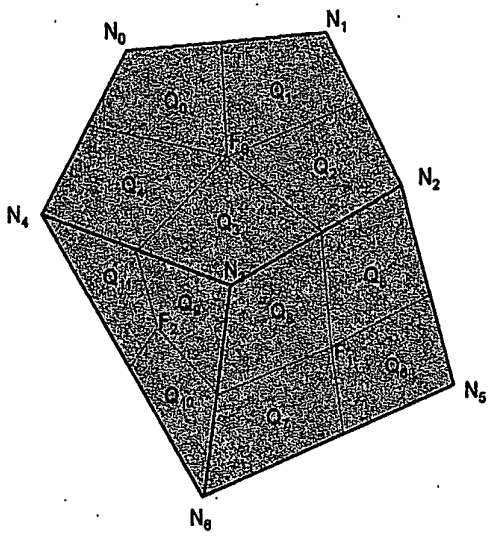


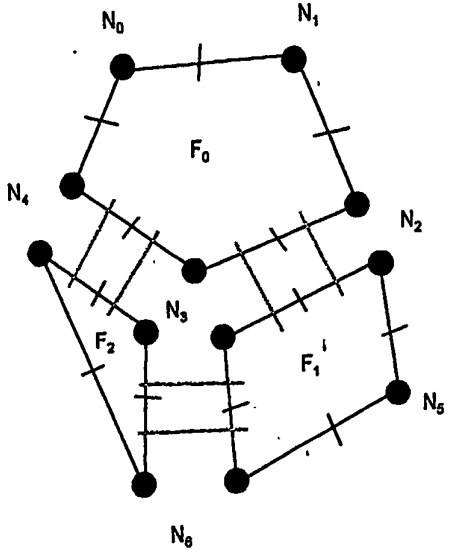
Fig. 12

```
Nodes 7
3 6  // N0
5 6  // N1
6 5  // N2
4 4  // N3
2 5  // N4
7 3  // N5
4 2  // N6

Faces 3
5 0 1 2 3 4  // F0
4 3 2 5 6    // F1
3 4 3 6      // F2

Level 0 12
0 0 0 4 0 4 0 0 0 // Q0
1 0 0 4 0 4 0 0 0 // Q1
2 0 0 4 0 4 0 0 0 // Q2
3 0 0 4 0 4 0 0 0 // Q3
4 0 0 4 0 4 0 0 0 // Q4
2 1 0 4 0 4 0 0 0 // Q5
5 1 0 4 0 4 0 0 0 // Q6
6 1 0 4 0 4 0 0 0 // Q7
3 1 0 4 0 4 0 0 0 // Q8
3 2 0 4 0 4 0 0 0 // Q9
6 2 0 4 0 4 0 0 0 // Q10
4 2 0 4 0 4 0 0 0 // Q11
```





Level 0 - Gmap

Fig. 13

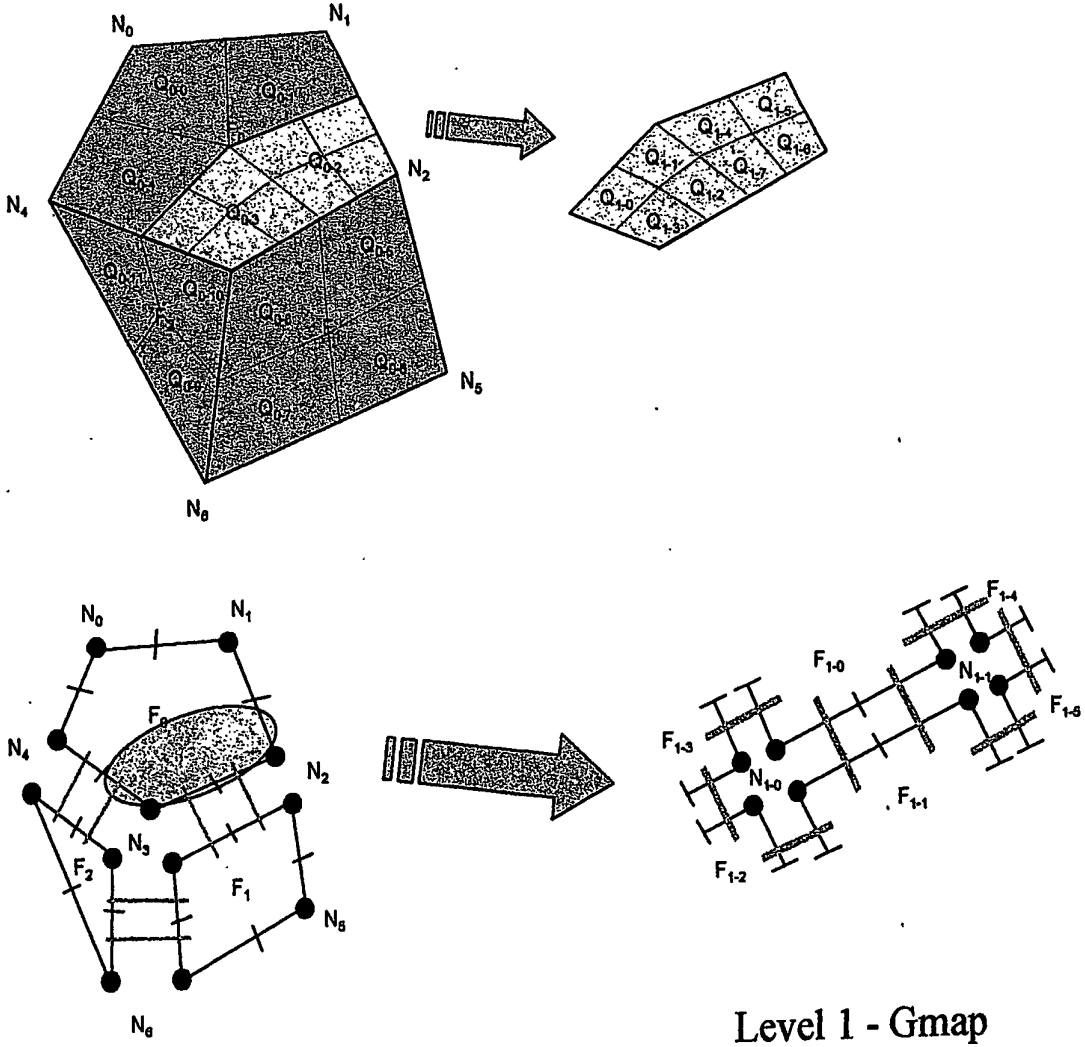
```
Nodes 7
3 6  // N0
5 6  // N1
6 5  // N2
4 4  // N3
2 5  // N4
7 3  // N5
4 2  // N6

Faces 3
5 0 1 2 3 4  // F0
4 3 2 5 6    // F1
3 4 3 6      // F2

Level 0 12
0 0 0 4 0 4 0 0 1  // Q0-0

4 2 0 4 0 4 0 0 1  // Q0-11

Level 1 8
3 0 0 4 0 4 0 0 0  // Q1-0
3 1 0 4 0 4 0 0 0  // Q1-1
3 2 0 4 0 4 0 0 0  // Q1-2
3 3 0 4 0 4 0 0 0  // Q1-3
2 4 0 4 0 4 0 0 0  // Q1-4
2 5 0 4 0 4 0 0 0  // Q1-5
2 6 0 4 0 4 0 0 0  // Q1-6
2 7 0 4 0 4 0 0 0  // Q1-7
```



Level 0 - Gmap

Level 1 - Gmap

Fig. 14